ATTORNEY DOCKET NO.
NVID-P000705

PATENT APPLICATION

TRANSPARENT ANTIALIASED MEMORY ACCESS

INVENTORS:

John S. Montrym
10451 Serra Street
Cupertino, CA 95014
U.S. Citizen

Brian D. Hutsell
655 Donahe Dr
Milpitas, CA 95035
U.S. Citizen

Steven E. Molnar
200 Perry Creek Dr
Chapel Hill, NC 27514
U.S. Citizen

Gary M. Tarolli
788 Strawberry Hill Rd.
Concord, MA 01742
U.S. Citizen

Christopher T. Cheng
1232 Loyola Drive
Santa Clara, CA 95051
U.S. Citizen

Emmett M. Kilgariff
2159 Paseo Del Oro
San Jose, CA 95124
US Citizen

Abraham B. de Waal
7112 Earlswood Ct
San Jose, CA, 95120
South African Citizen

ASSIGNEE: NVIDIA Corporation
2701 San Tomas Expressway
Santa Clara, CA 95050

WAGNER, MURABITO & HAO, LLP
Two North Market Street
Third Floor
San Jose, California 95113
Telephone (408) 938-9060

TRANSPARENT ANTIALIASED MEMORY ACCESS

By Inventors

John S. Montrym, Brian D. Hutsell, Steven E. Molnar, Abraham B. de Waal, Christopher T. Cheng, Emmett M. Kilgariff and Gary M. Tarolli

CROSS REFERENCE TO RELATED APPLICATIONS

This application claims priority from U.S. Provisional Patent Application No. 60/406,421 filed on August 27, 2002 and entitled "Transparent Antialiased Memory Access," which is incorporated herein by reference in its entirety.

BACKGROUND OF THE INVENTION

1. Field of the Invention

[1] This invention relates generally to computer graphics, and more particularly to transparent memory access using a virtual frame buffer to access frame buffer memory, e.g., super frame buffer memory.

2. Description of the Related Art

[2] When rendering an image, polygon edges that are not horizontal or vertical often appear jagged using conventional graphics systems because each pixel has finite physical dimensions. For example, since each pixel is essentially square, a diagonal line or edge rendered using the square pixels appears jagged, similar to a staircase. This effect is known as aliasing. To reduce aliasing, conventional graphics systems perform antialiasing, which helps reduce the effects that the physical dimensions of the pixels have on the appearance of objects being displayed. As a result of antialiasing, diagonal lines and edges appear smoother.

[3] One common antialiasing technique is supersampling, which typically is performed for a portion of the display, called a tile, or the entire display at a time. Broadly speaking, each pixel

in the tile or display is considered to be an M x N matrix of subpixels. Data for each polygon in the display is evaluated at each subpixel. Thus, the alpha value, color, and other data for the polygon can differ in, and is evaluated at, each subpixel.

[4] Figure 1 is a block diagram showing a conventional computer graphics system 100 that uses supersampling to perform antialiasing. As shown in Figure 1, the computer graphics system 100 shows a superbuffer 102 for storing a multi-sampled display image, a memory frame buffer 104 for storing the actual displayed image, and an application 106 that creates and edits the actual displayed image. As mentioned above, each pixel in the actual displayed image is considered to be an M x N subpixel matrix that is stored in a superbuffer 102.

[5] As is well known to those skilled in the art, current computer graphics standards often require that applications 106 have access to, and the ability to edit, images in the memory frame buffer 104. Hence, the conventional computer graphics system 100 needs to express the superbuffer 102 as a normal frame buffer, which can be accessed by the application 106. To this end, when the application 106 requests to access the memory frame buffer 104, the conventional computer graphics system 100 reduces the superbuffer 102 to the resolution of the normal memory frame buffer 104. Generally, this is done in manner similar to that described above, wherein the subpixels for each displayed pixel are sampled to determine the exact composition of the displayed pixel.

[6] After the application 106 finishes editing the memory frame buffer 104, the conventional computer graphics system 100 recreates the superbuffer by expanding the memory frame buffer 104. In particular, each pixel within the memory frame buffer 104 is copied to all the subpixels comprising the M x N matrix for the pixels. Unfortunately, the process of reducing and expanding the superbuffer 102 to provide frame buffer access to applications 106 causes undesirable effects to occur in the image, as described next with reference to Figure 2A.

[7] Figure 2A is a diagram showing a reduction and expanding pixel/subpixel process utilized in a conventional computer graphics system. As mentioned above, when the application requests to edit the memory frame buffer, the conventional computer graphics system reduces the superbuffer to the resolution of a normal frame buffer. For example, Figure 2A shows a subpixel matrix 200a corresponding to a pixel of the normal frame buffer. In this example, the subpixel matrix 200a is a 2 x 2 matrix having two black subpixels and two white subpixels.

[8] To reduce the superbuffer to the resolution of the normal frame buffer, the conventional computer graphics system examines the subpixels corresponding to each pixel of the normal frame buffer. The subpixels are then processed to generate the corresponding pixel. For example, in the subpixel matrix 200a, 50% of the subpixels are black and 50% of the subpixels are white. When processed, a pixel having 50% black and 50% white is generated, which results in a gray pixel 202.

[9] More particularly, a filter kernel generally is used to process the subpixels of the superbuffer during reduction. The filter kernel is a function that processes a predefined number of subpixels to produce each pixel of the normal frame buffer. It should be noted that often the filter kernel is configured to process pixels in excess of the number of pixels comprising each subpixel matrix of the superbuffer, which can provide enhanced antialiasing. However, as noted below, a filter kernel thusly configured often also enhances undesirable color "bleeding" effects.

[10] Once each pixel 202 of the normal frame buffer 104 has been created, the application can access and edit the contents of the frame buffer 104. In this manner, the application can edit or replace the displayed image that is stored in the frame buffer. Often, an application will only change the values of particular pixels within the frame buffer, leaving remaining pixels unaltered. For example, when displaying a menu screen an application generally only alters pixels of selected menu elements. As a result, many pixels of the frame buffer 104 can remain unaltered after the application has completed editing. In Figure 2A, for example, the gray pixel

202 has not been altered by the application, and thus continues to have the same value as it did when first reduced.

[11] When the application has completed accessing the normal frame buffer 104, the superbuffer is reconstructed by replicating the pixels of the normal frame buffer. Specifically, for each pixel of the normal frame buffer, the properties of the pixel are copied to each subpixel of the corresponding subpixel matrix of the super buffer. For example, in Figure 2A, the gray color value of pixel 202 is copied to each subpixel in the new matrix 200b.

[12] Unfortunately, although the application program did not alter pixel 202, all the subpixels of the corresponding new subpixel matrix 200b have changed to gray. As a result, image color values can change throughout the image although they are not altered by the application program. In graphics systems wherein the filter kernel is configured to process pixels in excess of the number of pixels comprising each subpixel matrix of the superbuffer, this effect is compounded. In particular, color values continue to "bleed" into one another each time the superbuffer is reduced. As a result, in these configurations, the image can appear to continuously blur over time.

[13] Figure 2B illustrates another prior art system that performs supersampling in which the subpixels are spread out over four separate frame buffer memories. Within this system, two separate graphics chips are used, 210a and 210b, and four separate and separately addressable frame buffer memories are used 230a-230d. Each graphics chip 210a-210b renders into only the two frame buffers that are connected with it, e.g., chip 210a interfaces with and renders into frame buffers 230a and 230b and chip 210b interfaces with and renders into frame buffers 230c and 230c. The four frame buffers 230a-230d make up the superframe buffer 200a of Figure 2A where there are four subpixels for each pixel of the displayed frame buffer. Each subpixel is stored in a respective frame buffer of buffers 230a-230d. When displaying an image on display 250, a combiner 240 (e.g., in the master device of the graphics chips) performs an aggregation function on the four subpixels from the frame buffers 230a-230d and

outputs a single value for the pixel. A processor 220 interfaces with chips 210a and 210b. In systems of Figure 2B that are supersampled, four Z values and four color values are stored for each pixel; that is each subpixel contributes a respective color and a respective Z value.

[14] In operation, when writing a triangle primitive into the frame buffers, each graphics chip would write into one of its frame buffers, e.g., 230a, and then jitter the primitive and write the jittered version into the second frame buffer, e.g., 230b. On read back, the subpixels from the frame buffers 230a-230d were aggregated as described above.

[15] A problem with the system of Figure 2B is that it is complicated because it requires four separate frame buffer memories to provide a superbuffer having four subpixels per frame buffer pixel. Each graphics chip can only render into two frame buffer memories. Therefore, if more subpixels are required per pixel, then more graphics chips would be required. Moreover, the aggregation function 240 is computationally expensive and the time required to separately address each of the four frame buffers 230a-230d can reduce graphics performance.

[16] In view of the foregoing, there is a need for systems and methods for antialiasing that provide enhanced antialiasing without undesirable side effects, such as color value changes and blurring. The methods should further allow application programs to access the buffer as required by current industry standards.

Summary of the Invention

[17] Generally, embodiments of the present invention fill these needs by `providing antialiasing using a frame buffer and a virtual frame buffer, which allow applications to access the frame buffer. In one embodiment, antialiased memory access includes receiving a request to access a memory address. The memory address is examined to determine if the memory address is within a virtual frame buffer. If the memory address is within a virtual frame buffer then the memory address is transformed into one or more physical addresses within a frame buffer that is utilized for antialiasing. A subpixel located at each of the one or more physical addresses within the frame buffer is then accessed. In one embodiment, the present invention provides for direct access by a software application. Supersampling or multisampling may be used.

[18] Supersampling is rendering a high resolution image, such that each fragment that is textured and shaded corresponds to a subpixel sample. Multisampling is similar, but each fragment corresponds to a pixel, but contains subpixel coverage. The (single) fragment color is written to all samples that are updated.

[19] Antialiased memory read access is provided in an additional embodiment of the present invention. As above, a request to read a memory address within memory is received. A determination is then made as to whether the memory address is within a predefined memory range. When the memory address is outside the predefined memory range, the data at the memory address is accessed normally. However, when the memory address is within the predefined memory range, the memory address is transformed into at least one physical address within a frame buffer utilized for antialiasing, and a subpixel value at the physical address within the frame buffer is read. Optionally, the processor, e.g., CPU, can be

provided with a pitch value of the frame buffer. In this aspect, the CPU can calculate a physical address within the frame buffer using the pitch value of the frame buffer.

[20] Additionally, antialiased memory read access is provided in a further embodiment of the present invention which operates as follows. A request to read a memory address within memory is received. A determination is then made as to whether the memory address is within a predefined memory range. When the memory address is outside the predefined memory range, the data at the memory address is read normally. However, when the memory address is within the predefined memory range, the memory address is transformed into a plurality of physical addresses within a frame buffer utilized for antialiasing. Each physical address stores a subpixel value related to a specific pixel. The plurality of subpixel values at the plurality of physical addresses within the frame buffer is read, and the subpixel values are combined to generate a pixel value for the specific pixel. In one aspect, the subpixel values are combined by blending the subpixel values into a single color value.

[21] Additionally, antialiased memory write access in a further embodiment of the present invention is described as follows. A request to write a data value to a memory address within memory is received, and determination is made as to whether the memory address is within a predefined memory range. When the memory address is within the predefined memory range, the memory address is transformed into a plurality of physical addresses within a frame buffer utilized for antialiasing, and the data value is written to the plurality of physical addresses within the frame buffer. As above, the data value can be written to the memory address normally when the memory address is outside the predefined memory range. Optionally, a base address of the predefined memory address can be the same as a base address of the frame buffer. However, this is not required, and in some embodiments the base address of the predefined memory address can be different from a base address of the frame buffer.

[22] In this manner, the processor, e.g., the central processing unit, can access pixel values generated from values in the frame buffer in a manner similar to accessing a normal frame buffer. Further, the subpixels stored in the frame buffer remain unaltered after read operations. Moreover, using the embodiments of the present invention, software copying from the frame buffer to a normal sized frame buffer is no longer required. As a result, performance is increased because clock cycles are no longer spent on copying, and quality is increased because non-requested subpixel color changes no longer occur.

[23] In another embodiment, a read access to antialiased memory includes receiving a request to read a memory address. In this embodiment, if the memory address is within a virtual frame buffer, then transforming the memory address into at least one physical address within a frame buffer that is utilized for antialiasing. A subpixel value is read from the at least one physical address within the frame buffer.

[24] Another embodiment, antialiased memory read access includes receiving a request to read a memory address and determining whether the memory address is within a virtual frame buffer. In this embodiment, if the memory address is outside the virtual frame buffer the data at the memory address is read. If the memory address is within the virtual frame buffer, the memory address is transformed into multiple physical addresses within a frame buffer that is utilized for antialiasing. Each physical address stores a subpixel value related to a specific pixel. The subpixel value is read from each of the multiple physical addresses within the frame buffer. The subpixel values can then be combined to generate a pixel value for the specific pixel.

[25] Another embodiment is drawn to antialiased memory write access including receiving a request to write a data value to a memory address. The memory address is examined to determine if the memory address is within a virtual memory buffer. When the memory address is within the virtual memory buffer, the memory address is transformed into multiple physical

addresses within a frame buffer that utilized for antialiasing. The data value is written to the multiple physical addresses within the frame buffer.

[26] Another embodiment describes reading a frame buffer where the frame buffer includes multiple pixels, each pixel includes multiple subpixels. In this embodiment, an address is received that corresponds to a pixel. The received address is transformed into at least one subpixel address and at least one subpixel is read from the frame buffer using at least one subpixel address. The read subpixel(s) can be blended to create a pixel value.

[27] Another embodiment describes reading a frame buffer where the frame buffer includes multiple pixels, each pixel includes multiple subpixels. An address is received. The received address is transformed into at least one subpixel address and at least one subpixel is read from the frame buffer using at least one subpixel address. The read subpixel(s) can be blended to create a pixel value. The created pixel value is supplied as if it were a pixel value at the received address.

[28] Another embodiment describes writing a frame buffer where the frame buffer includes multiple pixels, each pixel includes multiple subpixels. The embodiment includes receiving an address and a pixel value from a computer program. The computer program supplying the address and pixel value as if accessing a frame buffer that does not include subpixels. The received address is transformed into at least one subpixel address and the pixel value is written as at least two subpixel values using the at least one subpixel address.

[29] Another embodiment includes reading a frame buffer where the frame buffer includes multiple pixels, each pixel includes multiple subpixels. The embodiment includes receiving an address. The received address is transformed into a subpixel address and at least one subpixel value is read from the frame buffer using the subpixel address. The read subpixel value is supplied as if it were a pixel value at the received address.

[30] Another embodiment includes supplying a virtual frame buffer to a computer program including supplying a base address and buffer size information to the computer program where the base address and the buffer size information corresponds to a virtual frame buffer. An address is received in the virtual frame buffer from the computer program. The received address is transformed into at least one subpixel address where the subpixel address being an address in a frame buffer. At least one subpixel is read from the frame buffer using the subpixel address. If more than one subpixel is read, then the subpixels can be blended to create a pixel value. The created pixel value is supplied to the computer program as if it were a pixel value located at the received address in the virtual frame buffer such that the computer program does not directly access the frame buffer.

[31] Another embodiment includes supplying a virtual frame buffer to a computer program including supplying a base address and buffer size information to the computer program. The base address and the buffer size information corresponding to a virtual frame buffer. An address is received in the virtual frame buffer from the computer program. A pixel value is received. The received address is transformed into at least one subpixel address, the subpixel address being an address in a frame buffer. The pixel value is written as at least two subpixels values into the frame buffer using the subpixel address such that the computer program does not directly access the frame buffer.

[32] Other aspects and advantages of the invention will become apparent from the following detailed description, taken in conjunction with the accompanying drawings, illustrating by way of example the principles of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

[33] The invention, together with further advantages thereof, may best be understood by reference to the following description taken in conjunction with the accompanying drawings in which:

[34] Figure 1 is a block diagram showing a conventional computer graphics system that uses supersampling to perform antialiasing;

[35] Figure 2A is a diagram showing a reduction and expanding pixel/subpixel process utilized in a conventional computer graphics system;

[36] Figure 2B is a block diagram of a prior art system in which four subpixels comprise a pixel and each subpixel is stored in a separate frame buffer memory;

[37] Figure 3 is a block diagram showing a computer graphics system that uses supersampling and a virtual frame buffer to perform antialiasing, in accordance with an embodiment of the present invention;

[38] Figure 4 is a block diagram showing a read access operation using a computer graphics system in accordance with an embodiment of the present invention;

[39] Figure 5 is a block diagram showing a write access operation using a computer graphics system in accordance with an embodiment of the present invention;

[40] Figure 6 is a diagram illustrating a graphics memory configuration using a virtual frame buffer, in accordance with an embodiment of the present invention;

[41] Figure 7A illustrates virtual addresses for the pixel group of four horizontally adjacent pixels A, B, C, and D;

[42] Figure 7B illustrates generated physical addresses for a subpixel group of sixteen subpixels corresponding to the pixels of the pixel group;

[43] Figure 8 is a flowchart showing a method for performing a read operation using a virtual frame buffer and blending, in accordance with an embodiment of the present invention;

[44] Figure 9 is a flowchart showing a method for performing a read operation using a virtual frame buffer and subpixel selection, in accordance with an embodiment of the present invention; and

[45] Figure 10 is a flowchart showing a method for performing a write operation using a virtual frame buffer and subpixel selection, in accordance with an embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[46] A methods are disclosed for antialiasing using a frame buffer and a virtual frame buffer. In the following description, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art that the present invention may be practiced without some or all of these specific details. In other instances, well known process steps have not been described in detail in order not to unnecessarily obscure the present invention.

[47] Figures 1 and 2A, and Figure 2B were described in terms of the prior art. Figure 3 is a block diagram showing a computer graphics system 300 that uses supersampling and a virtual frame buffer to perform antialiasing, in accordance with an embodiment of the present invention. Supersampling is used to improve the perceived quality of the display by attenuating aliasing, even though the display resolution is fixed. In supersampling, a number of regularly spaced samples are taken for each pixel. These samples are then re-sampled or averaged out to yield the pixel value on display. This effectively corresponds to creating a virtual image at high resolution and digitally down-filtering the result, to eliminate the high frequencies that are the source of aliasing.

[48] Generally, the virtual frame buffer presents, in one embodiment, the illusion of a conventional frame buffer with regard to access by the application and other frame buffer clients, whereas the underlying implementations contain additional information required for antialiasing and support antialiased rendering and display and optionally could be compressed and not directly readable.

[49] As shown in Figure 3, the computer graphics system 300 utilizes a frame buffer 302 for storing a multi-sampled display image, a virtual buffer 304 for providing buffer access, and a processor, e.g., a central processing unit (CPU) 306, which executes the application

programs that create and edit the displayed image. As above, each pixel in the actual displayed image is stored as a number of subpixels in the frame buffer 302. In one embodiment, the subpixels are stored in an M x N matrix of subpixels in the frame buffer 302. Each subpixel can represent one sample of the pixel in the actual displayed image. The samples of the pixel in the actual displayed image can be from any portion of the pixel according to one or more of the well-known pixel sampling techniques.

[50] Many current computer graphics standards require that the CPU 306 have direct access to a frame buffer in order for application programs to have the ability to edit images in the frame buffer. Embodiments of the present invention provide such access using a virtual frame buffer 304. Broadly speaking, the virtual frame buffer 304 provides address mapping to the frame buffer 302, simulating direct buffer access to the CPU and the application program.

[51] Generally, when an application executing on the CPU 306 requests the ability to access the frame buffer, the hardware is initialized to configure the virtual frame buffer 304, and a pointer to the virtual frame buffer is returned to the application. The hardware intercepts each request within the virtual frame buffer 304, along with the associated frame buffer address. The hardware then maps the frame buffer address to one or more corresponding addresses in a frame buffer 302 that contain the corresponding one or more subpixels. The corresponding addresses may be adjacent in one embodiment. In this manner, the application can access the frame buffer 302 in a manner that simulates direct frame buffer access. However, as will be described subsequently, embodiments of the present invention advantageously avoid non-requested color changes to subpixels when the application program does not alter them. Furthermore, color "bleeding" is avoided using the embodiments of the present invention, as described in greater detail below.

[52] While the memory accesses are described in terms of accessing addresses for pixels and subpixels, one skilled in the art will understand that multiple bytes may be required to define each pixel and subpixel and therefore each address can represent multiple addresses of the multiple bytes. Each memory access can therefore access one or more pixels or subpixels or portions thereof.

[53] In one embodiment, an initialization of the computer graphics system includes defining a base address and a limit address of a virtual frame buffer. As an alternative to a limit address, a virtual frame buffer size can be defined. In addition, the pitch and bit depth of the virtual frame buffer are defined. The bit depth is the number of bits per pixel (e.g., 8-, 16-, 32-bits per pixel and other sizes). The pitch of the virtual frame buffer can be selected to match the pitch of the frame buffer but in alternative embodiments may not match the pitch of the frame buffer. A filter kernel is also defined to provide the down filtering for reading the frame buffer. The down filtering can use any filter scheme desired by the user. A pixel sampling scheme is also defined. The pixel sampling scheme defines the number of samples per pixel and the location and selection of the samples on each pixel that are then stored a subpixel values.

[54] Figure 4 is a block diagram showing a read access operation using a computer graphics system in accordance with an embodiment of the present invention. Although Figure 4 illustrates 2 x 2 sampling in the frame buffer, it should be noted that any size sampling can be utilized with the embodiments of the present invention. Further, it is appreciated that the samples may not be in a regular grid in other embodiments of the present invention and the regular grid shown is merely exemplary. Broadly speaking, embodiments of the present invention identify a region of graphics memory addresses in which requests from the CPU to access graphics memory will undergo a transformation. The transformation allows an application program executing on the CPU to function as though it is accessing a normal frame buffer, when in actuality the application program is accessing the subpixels of the frame buffer.

[55] To read a pixel from the frame buffer, the CPU 306 provides the address of the pixel in graphics memory. Embodiments of the present invention utilize the virtual frame buffer 304 to

simulate a normal frame buffer to application programs executing on the CPU 306. Thus, in the example of Figure 4, the CPU 306 can read a pixel using the virtual frame buffer 304 by supplying the address of the pixel. The memory address supplied by the CPU 306 will be referred to hereinafter as the "virtual address" 400 of the pixel.

[56] Once received, the virtual frame buffer 304 translates the virtual address 400 into one or more physical addresses 402 within the frame buffer 302. The number of physical addresses 402 that map to one virtual address 400 depends on the size of the subpixel matrixes used in the frame buffer 302. In particular, the number of physical addresses 402 that map to one virtual address 400 is equal to the number of subpixels included in each subpixel matrix used in the frame buffer 302. For example, in Figure 4 the frame buffer 302 uses 2 x 2 subpixel matrixes for antialiasing, each having a total four subpixels. Thus, four physical addresses 402 will be mapped to each virtual address 400 in the example of Figure 4.

[57] The virtual frame buffer 304 then performs a read of one or more of the subpixels at the physical addresses within the frame buffer 302. In one embodiment, the virtual frame buffer 304 performs the read by returning one of the subpixels at the physical address. In a further embodiment, the virtual frame buffer 304 performs the read by blending the subpixels at the physical addresses to generate the pixel value returned to the CPU 306. For example, in the subpixel matrix 402 of Figure 4, 50% of the subpixels are black and 50% of the subpixels are white. When blended, a pixel having 50% black and 50% white is generated, which results in a gray pixel 400. In this manner, the CPU 306 receives a pixel value for the pixel virtual address 400 supplied to the virtual frame buffer 304 in a manner similar to accessing a normal frame buffer. Further, the subpixels 402 stored in the frame buffer remain unaltered after the read operation. Similar operations are performed during a write operation, as described next with reference to Figure 5.

[58] Figure 5 is a block diagram showing a write access operation using a computer graphics system in accordance with an embodiment of the present invention. As above, although Figure 5 illustrates 2 x 2 sampling in the frame buffer, it should be noted that any size sampling could be utilized with the embodiments of the present invention.

[59] To write a pixel to the frame buffer, the CPU 306 provides the address of the pixel in the computer memory (e.g. the graphics memory or other portions of the computer memory) and the pixel value. As above, embodiments of the present invention utilize the virtual frame buffer 304 to simulate a normal frame buffer to application programs executing on the CPU 306. Thus, in the example of Figure 5, the CPU 306 can write a pixel using the virtual frame buffer 304 by supplying the address of the pixel, that is, the virtual address 500, and the pixel value.

[60] Once received, the virtual frame buffer 304 translates the virtual address 500 into one or more physical addresses 502 within the frame buffer 302. As during a read operation, the number of physical addresses 502 that map to one virtual address 500 depends on the size of the subpixel matrixes used in the frame buffer 302. In particular, the number of physical addresses 502 that map to one virtual address 500 is equal to the number of subpixels included in each subpixel matrix used in the frame buffer 302. For example, in Figure 5 the frame buffer 302 uses 2 x 2 subpixel matrixes for antialiasing, each having a total four subpixels. Thus, four physical addresses 502 will be mapped to each virtual address 500 in the example of Figure 5.

[61] The virtual frame buffer 304 then writes the provided pixel value to one or more of the subpixels at the physical addresses 502 within the frame buffer 302. In one embodiment, during a write operation, the pixel values provided to the virtual frame buffer 304 are replicated to all the subpixels of the corresponding subpixel matrix 502 in the frame buffer 302. However, it should be noted that embodiments of the present invention can further process the subpixels of the corresponding subpixel matrix 502 in the frame buffer 302 when additional image information is known. In such cases, varying subpixel values can be selected for the subpixels of the corresponding subpixel matrix 502 based on the provided pixel values and

surrounding pixel values. In this manner, the CPU 306 can write pixel values for the pixel to the virtual address 500 supplied to the virtual frame buffer 304 in a manner similar to accessing a normal frame buffer.

[62] Figure 6 is a diagram illustrating a graphics memory configuration using a virtual frame buffer 304, in accordance with an embodiment of the present invention. As shown in Figure 6, an embodiment of the present invention defines a region of graphics memory 600 as a virtual frame buffer 304. Once defined, requests to access the graphics memory 600 are examined to determine the memory address of the request. Graphics memory requests to access memory addresses before the beginning address 610 of the virtual frame buffer 304, or after the ending address 612 of the virtual frame buffer 304, are allowed to occur as normal. That is, the memory access request is allowed to occur without transformation.

[63] However, when an incoming address falls within the address range that defines the extent of the virtual frame buffer 304, the incoming address 602 is transformed into physical addresses 604 within the frame buffer 302. For example, when the incoming address is for a pixel 602 in the virtual frame buffer 304, the incoming address is transformed into four physical addresses 604 within the frame buffer 302, in the example of Figure 6.

[64] In one embodiment, a pitch for the frame buffer 302 is stored in a register, which is provided to the CPU when calculating memory addresses within the virtual frame buffer 304. The pitch is the distance, in address units, between two vertically adjacent pixels. When using the frame buffer 302, the pitch is the distance, in address units, between two vertically adjacent subpixels. As is well known in the art, when given an X-Y frame buffer, where X defines the width and Y defines the height, an address of a pixel can be defined as:

(1) (y * pitch) + (x * bytes per pixel) + base address of frame buffer,

[65] where x is the x-coordinate of the pixel, y is the y-coordinate of the pixel, and pitch is the pitch of the frame buffer. This calculation is generally performed using the CPU prior to a memory access request. Embodiments of the present invention allow the CPU to use the same equation when using the virtual frame buffer, however, an embodiment of the present invention can provide the CPU with the pitch of the frame buffer 302 as the pitch of the virtual frame buffer instead of providing the pitch of a normal frame buffer. In this manner, the CPU can assist the GPU in computing addresses within the frame buffer transparently as if it is calculating an address for a normal frame buffer.

[66] The base address used for the virtual frame buffer 304 and the frame buffer 302 can be configured to reduce area and cost. Specifically, the base address for the virtual frame buffer 304 and the frame buffer 302 can be the same to simplify the physical address computations. However, it should be noted that embodiments of the present invention are not limited to identical base addresses, and thus, the virtual frame buffer 304 and the frame buffer 302 can have different base addresses. Moreover, in some embodiments, the base address for the frame buffer 302 can be outside available memory space. For example, this virtual frame buffer may live in its own dedicated address space without underlying physical memory.

[67] For example, in one embodiment, when performing a write operation to a pixel 602 having a memory address within the virtual frame buffer 304, the pixel values are copied to all the corresponding subpixels 604 in the frame buffer 302. In particular, physical addresses are generated for each subpixel 604 of the subpixel matrix corresponding to the pixel 602. Then, the properties of the pixel 602 are written to each subpixel 604 of the subpixel matrix corresponding to the pixel 602.

[68] When performing a read operation, as above, physical addresses are generated for each subpixel 604 of the subpixel matrix corresponding to the pixel 602. The subpixels 604 are then read. In one embodiment, a filter that blends the subpixels 604 is applied and a resulting pixel value is returned to the CPU as the value for pixel 602. In a further embodiment, one subpixel 604 is selected from the subpixels corresponding to the pixel 602, and the value of the selected subpixel is returned to the CPU as the value for pixel 602. In one embodiment, only the subpixels contributing to the value of the pixel must be read.

[69] In addition, embodiments of the present invention allow any granularity access that the CPU can accomplish using a normal frame buffer, to map correctly to the super buffer 302. Thus, if the CPU can access multiple pixels at once, the virtual frame buffer 304 can transform the addresses of the pixels into physical addresses within the frame buffer 302. For example, when the CPU is capable of performing thirty-two bit memory accesses and there are eight bits per pixel, four pixels can be accessed simultaneously by the CPU. When the CPU accesses a pixel group 606 of four pixels in one memory access, the addresses of the pixel group 606 are transformed into sixteen physical addresses 608 within the frame buffer 302, in the example of Figure 6. As previously mentioned, although the examples discussed thus far have been described in terms of 4x data sampling for antialiasing, it should be noted that any size data sampling can be used with the embodiments of the present invention.

[70] Figures 7A and 7B illustrate a example of a thirty-two bit CPU access using a virtual frame buffer where each pixel is eight bits, in accordance with an embodiment of the present invention. Specifically, Figure 7A illustrates virtual addresses for the pixel group 606 of four horizontally adjacent pixels A, B, C, and D. When the virtual frame buffer receives pixel group 606, the virtual frame buffer transforms these virtual addresses into a plurality of physical addresses within the frame buffer, as shown in Figure 7B.

[71] Figure 7B illustrates generated physical addresses for a subpixel group 608 of sixteen subpixels corresponding to the pixels of the pixel group 606. In the example of Figure 7B, the frame buffer uses 2 x 2 subpixel matrixes to represent each pixel. Hence each virtual address of the pixel group 606 is transformed into four physical addresses within the frame buffer. For example, virtual address A is transformed into four physical addresses A_0, A_1, A_2, and A_3 in the frame buffer. Similarly, virtual address B is transformed into four physical addresses B_0, B_1, B_2, and B_3 in the frame buffer. A similar transformation occurs for virtual address C

and virtual address D. In each group of physical addresses A_0-A_3, B_0-B_3, C_0-C_3, and D_0-D_3, the pitch used is the stored pitch of the frame buffer, as described previously.

[72] Figure 8 is a flowchart showing a method 800 for performing a read operation using a virtual frame buffer and blending, in accordance with an embodiment of the present invention. Process 800 may be implemented as instructions stored in a memory of a computer system and executed on one or more processors thereof. As mentioned above, the virtual frame buffer can perform read operations by returning one of the subpixels at the physical address, or by blending the subpixels at the physical addresses to generate the pixel value returned to the CPU. Method 800 illustrates a read operation wherein the subpixels are blended to generate the pixel value returned to the CPU.

[73] In operation 804, a graphics memory address is received from the CPU. In one embodiment, a graphics processing unit (GPU) is utilized to facilitate access to graphics memory. Generally, the GPU is assigned its own graphics memory, which can include the virtual frame buffer, frame buffer, and other memory, such as command buffers. To access the graphics memory, the CPU sends a request to the GPU to access memory at a particular address, or range of addresses. In alternative embodiments, the CPU can also directly access the graphics memory and in particular the frame buffer.

[74] A decision is then made as to whether the memory address is within the defined address range of the virtual frame buffer, in operation 806. As mentioned above, embodiments of the present invention define a region of graphics memory as a virtual frame buffer, however the virtual frame buffer can also be in other portions of the memory other than the graphics memory. Once defined, requests to access the graphics memory are examined to determine the memory address of the request. If the memory address is outside the defined address range of the virtual frame buffer, the method 800 branches to operation 808. Otherwise, the method 800 continues with operation 810.

[75] In operation 808, the value at the received memory address is read and provided to the CPU. Hence, CPU requests to access memory addresses before the beginning address of the virtual frame buffer, or after the ending address of the virtual frame buffer, are allowed to occur as normal. That is, the memory access request is allowed to occur without transformation.

[76] However, if the memory address is within the defined address range of the virtual frame buffer, the virtual address is transformed into physical addresses within the frame buffer, in operation 810. Specifically, when performing a read operation, physical addresses are generated for each subpixel of the subpixel matrix corresponding to the pixel. In one embodiment, the CPU can assist in calculating the physical addresses within the frame buffer. In this embodiment, the pitch of the frame buffer is stored in a register. This pitch value is then provided to the application when access is requested to the frame buffer.

[77] The subpixels are read from the frame buffer, in operation 812. During operation 812, the values stored at the physical addresses generated in operation 810 are read from the frame buffer. As mentioned above, the frame buffer can store data for each pixel in the actual displayed image as an M x N subpixel matrix. In this manner, the color values for the subpixels of the subpixel matrix can be varied to provide antialiasing.

[78] In operation 814, the subpixel values are filtered to generate a pixel value. As mentioned above, each pixel is stored as an M x N subpixel matrix in the frame buffer. In operation 814, these subpixels are filtered, or "blended," to provide a color value for a pixel. For example, if 50% of the subpixels are black and 50% of the subpixels are white, when blended, a pixel having 50% black and 50% white is generated, which results in a gray pixel. However, it should be understood that this filter is not limited to an unweighted average in alternative embodiments the filter could use a weighted average or any other suitable filtering functions that are well known in the by those skilled in the art. The generated pixel value is then provided to the CPU, in operation 816 and the method operations end in operation 818.

[79] In this manner, the CPU receives a pixel value for the pixel virtual address supplied to the virtual frame buffer in a manner similar to accessing a normal frame buffer. Further, the subpixels stored in the frame buffer remain unaltered after the read operation. Moreover, using the embodiments of the present invention, software copying from the frame buffer to a normal sized frame buffer advantageously is no longer required. As a result, performance is increased because clock cycles are no longer spent on copying, and quality is increased because non-requested subpixel color changes no longer occur.

[80] Figure 9 is a flowchart showing a method 900 for performing a read operation using a virtual frame buffer and subpixel selection, in accordance with an embodiment of the present invention. Process 900 may be implemented as instructions stored in a memory of a computer system and executed on one or more processors thereof. As mentioned above, the virtual frame buffer can perform read operations by returning one of the subpixels at the physical address, or by blending the subpixels (e.g., through a filter) at the physical addresses to generate the pixel value returned to the CPU. Method 900 illustrates a read operation wherein one subpixel value is selected to generate the pixel value returned to the CPU.

[81] In operation 904, a graphics memory address is received from the CPU. In one embodiment, GPU is utilized to facilitate access to graphics memory. As mentioned previously, the GPU is assigned its own graphics memory, which can include the virtual frame buffer, frame buffer, and other memory, such as command buffers. To access the graphics memory, the CPU sends a request to the GPU to access memory at a particular address, or range of addresses.

[82] A decision is then made as to whether the memory address is within the defined address range of the virtual frame buffer, in operation 906. If the memory address is outside the defined address range of the virtual frame buffer, the method 900 branches to operation 908. Otherwise, the method 900 continues with operation 910.

[83] In operation 908, the value at the received memory address is read and provided to the CPU. Hence, CPU requests to access memory addresses before the beginning address of the virtual frame buffer, or after the ending address of the virtual frame buffer, are allowed to occur as normal. That is, the memory access request is allowed to occur without transformation.

[84] However, if the memory address is within the defined address range of the virtual frame buffer, the virtual address is transformed to a physical address within the frame buffer, in operation 910. Specifically, when performing a read operation, a physical address is generated for a subpixel of the subpixels corresponding to the pixel. In one embodiment, the CPU can assist in calculating the physical addresses within the frame buffer. In this embodiment, the pitch of the frame buffer is stored in a register. This pitch value is then provided to the application when access is requested to the frame buffer.

[85] A subpixel is selected from the frame buffer based on predefined selection criteria, in operation 912. As mentioned above, the frame buffer stores data for each pixel in the actual displayed image as an M x N subpixel matrix. In this manner, the color values for the subpixels of the subpixel matrix can be varied to provide antialiasing. In operation 912, a subpixel from the M x N subpixel matrix corresponding to the pixel is selected. For example, the subpixel sample nearest the center of the pixel in the subpixel matrix can be selected. The selected subpixel value is then read from frame buffer and provided to the CPU, in operation 914 and the method operations end in operation 916.

[86] In this manner, the CPU receives a pixel value for the pixel virtual address supplied to the virtual frame buffer in a manner similar to accessing a normal frame buffer. In addition, the embodiment of method 900 can increase performance because clock cycles and bandwidth needed to read unused subpixel values are saved.

[87] Figure 10 is a flowchart showing a method 1000 for performing a write operation using a virtual frame buffer, in accordance with an embodiment of the present invention. In operation 1004, a graphics memory address is received from the CPU. As mentioned previously, a GPU can be utilized to facilitate access to graphics memory. In this aspect, the GPU is assigned its own graphics memory, which can include the virtual frame buffer, frame buffer, and other memory, such as command buffers. To access the graphics memory, the CPU sends a request to the GPU to access memory at a particular address, or range of addresses.

[88] A decision is then made as to whether the memory address is within the defined address range of the virtual frame buffer, in operation 1006. If the memory address is outside the defined address range of the virtual frame buffer, the method 1000 branches to operation 1008. Otherwise, the method 1000 continues with operation 1010.

[89] In operation 1008, the value at the received memory address is written to the memory address provided by the CPU. Hence, CPU requests to access memory addresses before the beginning address of the virtual frame buffer, or after the ending address of the virtual frame buffer, are allowed to occur as normal. That is, the memory access request is allowed to occur without transformation.

[90] However, if the memory address is within the defined address range of the virtual frame buffer, the virtual address is transformed to physical addresses within the frame buffer, in operation 1010. Specifically, when performing a write operation, physical addresses are generated for each subpixel of the subpixel matrix corresponding to the pixel. In one embodiment, the CPU can assist in calculating the physical addresses within the frame buffer. In this embodiment, the pitch of the frame buffer is stored in a register. This pitch value is then provided to the application when it requests access to the frame buffer.

[91] The received pixel value is replicated to the subpixels at the generated physical addresses, in operation 1012. As mentioned above, the frame buffer stores data for each pixel in the actual displayed image as an M x N subpixel matrix. In this manner, the color values for the subpixels of the subpixel matrix can be varied to provide antialiasing. Thus, in operation 1012, the received pixel value is replicated to the subpixels of the subpixel matrix corresponding to the received virtual memory address. However, it should be noted that embodiments of the present invention can further process the subpixels of the corresponding subpixel matrix in the frame buffer when additional image information is known. In such embodiments, varying subpixel values can be selected for the subpixels of the corresponding subpixel matrix based on the provided pixel value and surrounding pixel values. The method operations end in operation 1016. In this manner, the CPU can write a pixel value for the pixel to the virtual address supplied to the virtual frame buffer in a manner similar to accessing a normal frame buffer.

[92] Although an MxN subpixel matrix is described above, it is appreciated that embodiments of the present invention apply to samples on an irregular grid. In these embodiments, there are N samples at locations in the neighborhood of the pixel center. Other representations are possible, aside from a regular pixel matrix, such as a set of fragments with subpixel coverage, hierarchies of samples, etc.

[93] It should be appreciated that the instructions represented by the operations in Figures 8-10 above are not required to be performed in the order illustrated, and that all the processing represented by the operations may not be necessary to practice the invention. Further, additional processing may precede or follow the operations described in Figures 8-10 above.

[94] Although the foregoing invention has been described in some detail for purposes of clarity of understanding, it will be apparent that certain changes and modifications may be practiced within the scope of the appended claims. Accordingly, the present embodiments are to be considered as illustrative and not restrictive, and the invention is not to be limited to the details given herein, but may be modified within the scope and equivalents of the appended claims.